A Network Scheduling Model for Distributed Control Simulation

Dennis Culley[*]
NASA Glenn Research Center, Cleveland, Ohio, 44135

George Thomas[†]
N & R Engineering, Cleveland, Ohio, 44135

Eliot Aretskin-Hariton[‡]
NASA Glenn Research Center, Cleveland, Ohio, 44135

Distributed engine control is a hardware technology that radically alters the architecture for aircraft engine control systems. Of its own accord, it does not change the function of control, rather it seeks to address the implementation issues for weight-constrained vehicles that can limit overall system performance and increase life-cycle cost. However, an inherent feature of this technology, digital communication networks, alters the flow of information between critical elements of the closed-loop control. Whereas control information has been available continuously in conventional centralized control architectures through virtue of analog signaling, moving forward, it will be transmitted digitally in serial fashion over the network(s) in distributed control architectures. An underlying effect is that all of the control information arrives asynchronously and may not be available every loop interval of the controller, therefore it must be scheduled. This paper proposes a methodology for modeling the nominal data flow over these networks and examines the resulting impact for an aero turbine engine system simulation.

I. Introduction

FOR a generation, Full Authority Digital Engine Control (FADEC) has been standard for most gas turbine engine applications because of its well-known performance advantages[1]. FADEC is a flexible software-driven strategy that integrates operational control across the entire engine system. This control function is reliant on the capability of electronics, which are a primary material of any modern, realizable control system. Protecting the electronics in a single enclosure is the hallmark of a centralized architecture. Distributed Engine Control (DEC) is a revolutionary change in the hardware architecture of a FADEC control system.[2,3,4,5,6] DEC provides the FADEC functionality using a flexible hardware-driven strategy that reduces system constraints on controls, as well as on the engine system.[7] A key enabler to the DEC technology is high temperature electronics.[8,9,10,11]

Although electronics provide the benefits in modern control systems, they are also a key weakness in any weight-limited application, such as the aero engine. Regardless that Moore's Law relentlessly increases electronic capabilities while reducing power consumption and volume; the limiting factor of electronics will always be reliability as a function of temperature.[11] This concern is what drives the centralized architecture and the need to restrict controller location, or modify the controller's local thermal environment, all of which have significant weight impacts due to wiring, packaging, and ancillary support systems. The objective of DEC has been to partition the control system hardware, in order to maximize the benefits. The main control law processing hardware should be able to take full advantage of rapidly evolving mainstream electronics technology, while simultaneously minimizing the weight penalty driven by the engine thermal environment. This desire is driving the development of a subset of high temperature electronics that can survive near the engine core with minimal weight impact. These embedded high temperature electronics result in what is known as a smart node, whose functionality helps to offset some of the burden on the controller. Meanwhile, the controller hardware can become more focused on the high processing function of

[*] Senior Research Engineer, dennis.e.culley@nasa.gov
[†] Controls Engineer, george.l.thomas@nasa.gov
[‡] Research Engineer, eliot.d.aretskin-hariton@nasa.gov

the control law and repositioned so that it no longer needs to be encumbered by a harsh thermal environment due to the previous constraints.

The control architecture described above is only realizable if information is sufficiently free to flow between the distributed hardware elements. Using digital network technology potentially resolves this issue by using a low weight wire harness with a common interface at each hardware element. The trade-off, however, is a sequential flow of digital data between these elements, leading to an inherent skew in the arrival time of information to and from the controller. Potentially, there is insufficient time to transfer the data.[12,13,14] Additionally, serial communication has no inherent mechanism to synchronize the transmitted information, which previously occurred by simultaneous sampling of continuous analog signals in a centralized architecture. This leads directly to the need to understand and model the effect of the sequential flow of data in a distributed control simulation. The purpose of a communication schedule model is to simulate the nominal exchange of information, i.e., non-fault condition, that occurs between the hardware controller and control elements. This is inherently a different problem than trying to analyze fault conditions like stability under packet loss,[15] which is not considered here.

In the near term, relative to a traditional centralized control architecture, DEC technology can provide aerodynamic and thermodynamic engine system benefits, improve engine life cycle costs, and reduce the total engine system weight. These near term objectives tend to focus on overcoming existing system constraints. For example, i) DEC hardware can be more easily configured to reduce engine drag by limiting the nacelle diameter, ii) high temperature electronics can improve heat quality, rejecting heat at higher temperature for more efficient cooling, while also being more easily located on the engine, iii) reusable modular components can reduce design and manufacturing costs, iv) and significant weight reduction can be achieved via a lighter wiring harness. In the long term, DEC technologies are a platform for introducing new control-based capabilities that will make significant contributions to engine performance, efficiency, and safety. For example, i) embedded signal processing in smart nodes can extract new information from each existing sensor location, ii) the simplified interface at the controller can enable faster redesign and access to state-of-the art computational electronics and, iii) in general, more processing capability improves the accuracy of system information and control. The discovery of additional control applications enabled by distributed control is beyond the present scope of this paper; however, we can assume that future control applications will place more burden on communications, leading us to consider methods for conserving throughput in our research.

The focus of this paper addresses the most significant functional impact of distributed control technology, namely the serial communication of data using network(s) between elements of the closed-loop control system. The objective is to define a modeling technique for simulation of the nominal data flow that is representative of distributed control on the turbine engine system. Section II explains the method for developing the network schedule model and how it will be applied to an engine system simulation, in this case the Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k).[16] Section III will demonstrate results for simulations using the communication schedule relative to the baseline C-MAPSS40k that performs full data transfer on every time step. Section IV will provide conclusions.

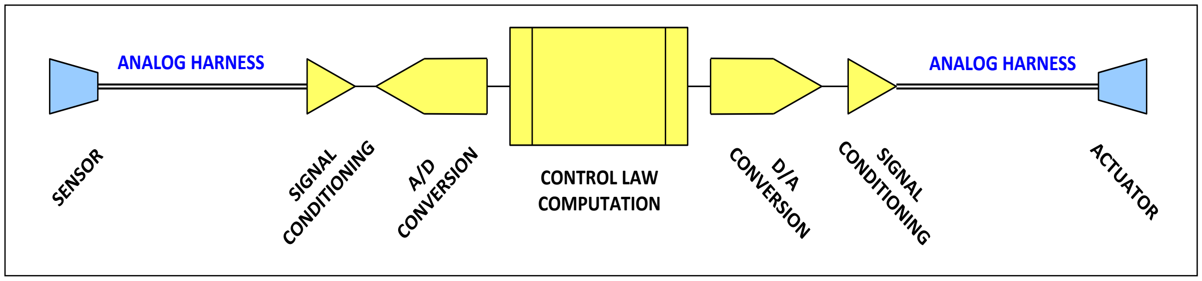
II. Modeling Distributed Control Networks

A. Hardware Effects

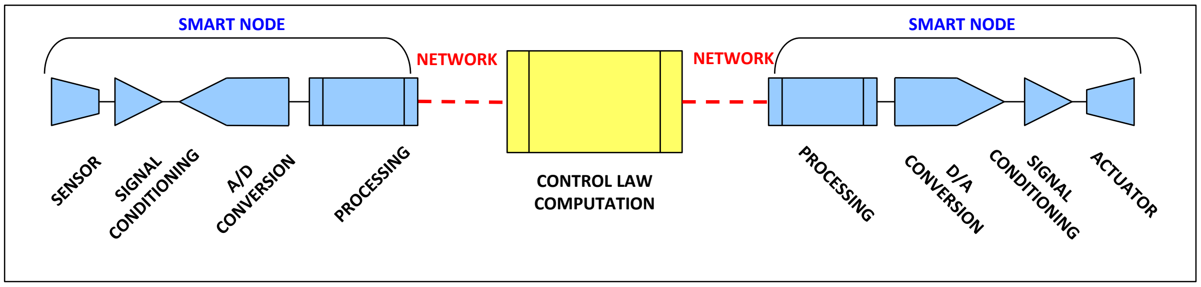
The immediate purpose of distributed control technology is not to change the function of the control system; rather, its initial purpose is to alleviate hardware implementation constraints due to an evolving turbine engine. That being said, by resolving the constraints imposed by hardware, a pathway to improved control system functionality becomes available. The intent of modeling the distributed control system is to account for and understand the effects of the hardware implementation, which have often been neglected.

Figure 1 compares the centralized (Fig. 1a) and distributed (Fig. 1b) control architectures by notionally describing the functional similarity in information flow through a very simplified control system. In both cases, on the far left is the transducer that converts a measureable physical parameter in the plant process to an electrical signal. The information obtained by the sensor is ultimately provided to the control law algorithm, which decides what action to take in response to the input. The control response is provided to a second transducer on the far right, which creates a physical change in the plant, altering its state. Note that the sensor and actuator require physical interfaces to the measured quantity and control surface within the plant. Their function is dependent on location. In contrast, the control law processing function only requires an electronic interface to the transducers. Therefore, the location of control law processing hardware is generally determined by weight optimization, for which the thermal environment and the wire harnessing weight per unit length are major factors.

Figure 1 also highlights the hardware differences between centralized control versus the distributed control implementation. The hardware components are nearly identical except the latter has an additional processor attached



a) **Centralized architecture. All electronics are contained in the engine control unit hardware, depicted in yellow. Information exchanged between the controller and its sensors and actuators exist as continuous time analog signals that are digitized at the controller.**



b) **Distributed architecture. High temperature electronics are embedded with the individual sensor and actuator hardware, which enables digitization at the source and data transmission over networks. The separate engine controller hardware, shown in yellow, contains no analog circuits, performing only control law processing.**

Figure 1. Notional information flow from sensor to controller to actuator for the (a) centralized architecture and (b) the distributed architecture. The blocks symbolize the functions being performed.

to each transducer “smart node.” The inclusion of the processor in the smart node allows it to perform its function independently from the controller. In the minimal form, the smart node converts signals between analog and digital domains and provides the means to communicate with the controller over the network. If the simple diagram in Figure 1a were expanded to describe an entire engine control system, the analog signaling interfaces at the centralized control unit would be very complex due to the number and wide variation in transducer types. The communication network avoids this physical wiring complexity by containing these differences within the smart node. Therefore, the distributed system has far more flexibility due to its low harness weight and simplified interface.

A significant difference between the control architectures occurs when considering the increased time it takes for the distributed architecture to transfer data between the smart node and controller, relative to the centralized case. Analog-to-digital and digital-to-analog conversions occur rapidly, on the order of microseconds. In the centralized case, these conversions are synchronized from a common clock and made available to the control law simultaneously. Other than the capability of the conversion hardware and the internal processor bus transfer speed, there is no other significant hardware restriction on the frequency of data input/output from the control law. A conservative point of reference for parallel 16-bit A/D or D/A conversion rate is 5 microseconds, and nearly negligible time for data transfer on the internal processor bus.

In contrast, the conversion speed in the smart nodes is similar but the data is transferred serially through a network to the control law computer. The same 16-bit value transferred through a serial network at 10 Megabits per second requires 1.6 microseconds, after the conversion. Unfortunately, serial communication protocol always requires additional information, greatly expanding the number of bits being transferred per message. A more realistic, yet conservative estimate of the message size would be 18 bytes requiring 14 microseconds at the 10 Mbps speed. However, other factors are to be considered: smart nodes must share serial network media; mechanisms like master/slave (command/response) protocol are used to avoid message collisions; accounting for message jitter and dead time between transfers; and lower bandwidth due to high temperature electronics. This easily can require many

10’s of milliseconds to complete the network data transfer for all the nodes. If the control loop interval cycles every 20 milliseconds, (50Hz) it becomes obvious that the network data transfer time becomes a potentially serious issue.

B. Decoupling Network Communication Rates from Control

A traditional approach to modeling the control system assumes complete transfer of all information between controller and smart nodes at the start of each control loop interval, which is consistent with the centralized control hardware architecture. Neglecting this data transfer time ignores one of the major implications of implementing a distributed system. However, requiring that complete information transfer occur potentially institutes a race condition where the data transfer and control law evaluation are competing for the same limited time interval. In the very least, it is likely that the controller would have to accumulate system sensor data during one interval, then evaluate the control law and command the actuators in the following interval. Still, this implies a significant dependence on the function of the hardware and potential complications for control stability that need to be understood. As an alternative, an assumption is made that it is not necessary to transfer all the data every control interval, potentially alleviating this hardware dependence. The objective then becomes determining a basis for the reduced frequency of network data transfer and testing the results. Recognizing that every network is a resource with finite bandwidth, a secondary objective is to efficiently move information and conserve available throughput in the anticipation of future growth.

The control loop interval time is based on the need to respond to the relevant plant dynamics. These phenomenon are unlikely to be dependent on any one system state or measurement, but to the overall complex interaction of the various plant components. In other words, the control loop interval is not based on the rate of change of any individual measured variable, but on the relative rates of change of all the plant variables with respect to each other. Our assumption is that in centralized control these sensor input values are highly oversampled by the controller. Therefore, some limitation of the sensor update rates in a distributed control architecture does not change the control law as long as we are not significantly altering the frequency content of the input signals. Knowledge of the plant dynamics is used as a basis for determining the rate to communicate information to the controller. This is the same process used to select a transducer, whose finite time response limits the frequency content of the signal from the plant. A similar argument can be made for the ability of the actuators to respond to commands from the controller.

The C-MAPSS40k engine system simulation provides a controller, a sensor suite, and an actuator suite. These are shown in Table 1. The control loop interval is 15 milliseconds. The sensor suite consists of a set of pressure, temperature, and shaft speed sensors, for which the time constants are expressed as an angular frequency. For pressure, each sensor is assumed to have a time constant value of 25 radians per second. Similarly, all temperature sensors are assumed to have time constants of 9 radians per second. Time constant information about the actuators in C-MAPSS40k was similarly assigned. All are shown in Table 2.

The shaft speed is determined by a pulse generated once every revolution, which is a very high rate. However, we are interested in the rate of change in shaft speed; therefore, the time constant of the information of interest will be a function of shaft inertia. In order to avoid a complex derivation for this value, we can rely on the relationship of fan speed to the thrust, which is valid regardless of the basis of the control law. A very simple shaft speed rate of change approximation then is to use the FAA requirement for thrust response as described in Csank, et al.[17] The requirement states that the engine thrust must be able to increase from less than 15% of its rated thrust to 95% of its

Table 1. Sensor and actuators in C-MAPSS40k

Wf	Fuel Flow Actuator
VSV	Variable Stator Vane Actuator
VBV	Variable Bleed Valve Actuator
P2	Pressure, Engine Inlet
P25	Pressure, High Speed Compressor Inlet
P30	Pressure, High Speed Compressor Outlet
P50	Pressure, Turbine Outlet
T2	Temperature, Engine Inlet
T25	Temperature, High Speed Compressor Inlet
T30	Temperature, High Speed Compressor Outlet
T50	Temperature, Turbine Outlet
Nc	Speed, Core Shaft
Nf	Speed, Fan Shaft

Table 2. Information provided or derived from the engine system simulation used to estimate a reduced network data rate, based on control element time constant, for information exchange with the controller.

Parameter	**time constant**			**Original over-sample**
	rad/sec	**Hz**	**sec**	
Control interval	419	66.7	0.015	1
Pressure	25	4.0	0.2513	17
Temperature	9	1.4	0.6981	47
Speed	6	1.0	1.0000	67
VSV	31	5.0	0.2000	13
VBV	31	5.0	0.2000	13
Wf	20	3.2	0.3121	21

rated value within a 5 second window. Therefore, we can assume the time constant for shaft speed rate of change is one fifth of this value, or 1 second.

C. Developing and Implementing the Network Schedule Model

The engine system simulation ideally replicates the physical system, that is, as a separate plant and control system. Furthermore, decomposition of the distributed control system is modeled as a suite of sensors, a controller, and a suite of actuators. This approach yields a set of processes that can operate independently.[18] In fact, these modular models could operate internally at different rates, faster or slower, as long as the interfaces are properly established. This is analogous to a system's physical implementation where the separate hardware elements bear no relationship with one another except for the virtue of being connected together. During each interval of the closed loop control simulation of this engine system, the sensors inform the control law of current plant conditions, which subsequently command the actuators manipulating the plant. This is shown in Figure 2.

The purpose of the plant model is to emulate the dynamics of the continuous system; therefore, the defined plant update rate must appropriately capture the relevant system dynamics. In reality, the plant has much faster dynamics, especially considering the local spatial details within components; however, those details do not necessarily factor into system level control and are generally not modeled for control development purposes.[19] For instance, C-MAPSS40k is a 0-dimensional dynamic simulation. The control law update rate is selected to insure system stability, but typically matches the plant model rate for convenience.

The sensors must capture the local transients of the physical parameter being measured in the plant, but this is not necessarily the rate for which information needs to be communicated to the control law. The implication is that each distributed node must respond to two rates, that of the plant and that of the control law's demand for information. This is more easily understood considering that future sensors may be used to evaluate wide bandwidth signals from the plant that could be used to extract more subtle indications of the plant state. In that case, the local processing capability of the smart node would be used to process and reduce the data measured at the plant in order to extract the relevant information, which would subsequently be reported to the controller at a slower rate. A similar situation exists with the actuators that must operate at a rate sufficient to dampen out engine transients, while also performing local loop

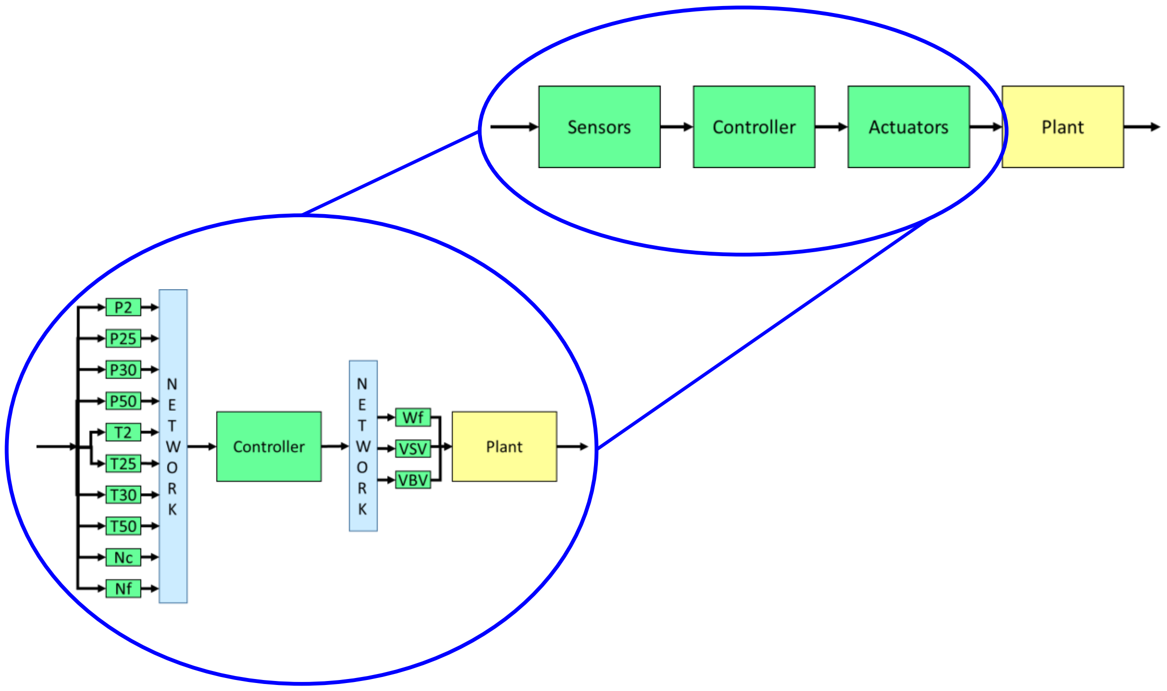


Figure 2. Decomposition of modular model structure based on distributed hardware implementation and sequential process execution during one simulation interval. Each model has the potential to operate at a different rate. The network model regulates information flow between the controller and control elements at a selectable rate.

closure. In essence, the smart nodes can perform a rate transition by using their inherent processing capability. Note that modeling a system to this level of detail requires multi-rate simulation capability.

In effect, distributed control systems implement a form of hierarchical control, where the main control law provides system wide coordination and the smart nodes execute specific localized functions. This describes the nature of distributed control; it is a system of asynchronous systems. Each smart node, by virtue of an embedded processor, operates from its own local time base. The communication network is the mechanism that provides the system synchronization, either purposely, or through virtue of its operation. The network model will restrict the data flow between the control elements and the controller based on a desired rate of data transfer as dictated by the engineering requirement. Subsequently, accommodations must be made for the control law to operate at every time step on the best available information, which means holding the last reported datum from any particular input parameter. The operation of the network is inherently a function of the controller.[20]

To accommodate these network effects in the system model requires a scheduling algorithm that regulates the flow of data between any control functions connected by a network. Two constructs are created as a basis for the model. The *major frame* defines the overall time interval for periodicity of the network data transfer process. It consists of multiple *minor frames* that define the minimum interval for data transfer to the control law. The control law is assumed to run in a continuous "while" loop, where the data made available at the start of the algorithm is operated upon. In our simulation, the exact arrival time of any data within the control loop interval is unknown, but is assumed current and arriving concurrently with the control law execution. It is queued, so as not to create the race condition. For modeling purposes, the control loop interval time is equivalent to the minor frame time of the network. Any skew in the data arrival time within this minor frame interval, (a requirement imposed by serial data transfer), is ignored.

For simplicity purposes, the major frame is further defined as 100 times the minor frame interval. This provides a convenient number of options for defining the periodicity of individual network data transfers. That is, periodicity is achieved by defining individual data transfers at any of the following multiples of the minor frame rate: 1, 2, 4, 5, 10, 20, 25, 50, or 100. Furthermore, because these intervals guarantee periodicity, individual smart nodes can be scheduled to begin their data transfer with some flexibility within the major frame. In effect, the communication schedule can be viewed as a gate, where data transfer between a smart node and the controller can only occur during the approved intervals regardless of what processing may be happening at the local element node.

The next issue is determining the frequency of individual data transfers within each major frame. Recall that previously these transfers occurred at every control interval. The result was that many data elements were highly oversampled. Using a schedule, we can control the degree of oversampling based on a more reasonable criteria. The chosen criteria is the time constant of each particular datum.

A simple first order system responds to a step input based on

$$u(t) = 1 - e^{-(t/\tau)}$$

where τ is the time constant. Three time constants are required to track a step response to the 95% level, five provides better than 99% accuracy. The objective is to choose a periodic minor frame interval for each datum that *at least* matches the time response of the transducer. For example, if a sensor has a time constant of 0.150 seconds, then it is desired to have that device sampled at least 5 times over a period of 0.750 seconds. Assuming the control interval time is 0.015 seconds, then the minor frame is also 0.015 seconds and the major frame is 1.500 seconds. Therefore, the example sensor should be sampled at least 10 times over the major frame interval. For this example device, this represents a reduction in data transfer to the controller of 90%.

Table 3 describes the time constants of each smart node device in our simulation and the selection of its appropriate network schedule interval (line 3 in the Table, labeled selected periodic interval).

The last aspect of setting up the schedule model is to try to balance the throughput on the network. Recall that distributed control requires serial data transfer over communication networks. Serial data transfer implies that only

Table 3. Derivation of the Basic Schedule for Communication Model. For each control element the time constant is provided in seconds, as well as an integer multiple of the minor frame rate at which the controller operates. This indicates the original degree of oversampling. A periodic interval rate is selected to make the communication rate periodic. Finally, an interval offset is introduced to spread the data transfers for minimum throughput.

	Controller	Wf	VSV	VBV	P2	P25	P30	P50	T2	T25	T30	T50	Nc	Nf
time constant (sec)	0.0150	0.3121	0.2000	0.2000	0.2513	0.2513	0.2513	0.2513	0.6981	0.6981	0.6981	0.6981	1.0000	1.0000
multiples of minor frame	1	20	13	13	16	16	16	16	46	46	46	46	66	66
selected periodic interval		10	10	10	10	10	10	10	25	25	25	25	50	50
starting interval offset		0	1	2	3	4	5	6	17	18	19	20	7	8

one device can transmit at any given time and every transmission requires a finite amount of time to be transmitted. Throughput is a system resource that must be properly allocated because, like every finite resource, requirements creep tends to consume all available resources. The criterion for balancing the schedule is to minimize throughput during every minor frame. This is accomplished by offsetting the minor frame position of each datum within the larger major frame while still maintaining the periodicity. These offsets are shown in the last row of Table 3.

The first 20 minor frames of the resulting schedule are shown in Table 4. Throughout the schedule there is typically only one message transferred per minor frame. The largest number of smart nodes that the control system will communicate with in a particular minor frame is two. For instance, during interval 20, the controller will send a command value to node Wf and read the sensor value from node T50. Occasionally, no messages are sent. It is useful to compare the network schedule implemented in Table 4 to the highly oversampled baseline case where all nodes communicate with the controller every minor frame. The network schedule shown in Table 4 reduces network messaging by 93.1% of the baseline case.

Table 4. The first 20 minor frames of the communication schedule.

	Communication Schedule												
minor frame	**Wf**	**VSV**	**VBV**	**P2**	**P25**	**P30**	**P50**	**T2**	**T25**	**T30**	**T50**	**Nc**	**Nf**
0	Wf												
1		VSV											
2			VBV										
3				P2									
4					P25								
5						P30							
6							P50						
7												Nc	
8													Nf
9													
10	Wf												
11		VSV											
12			VBV										
13				P2									
14					P25								
15						P30							
16							P50						
17								T2					
18									T25				
19										T30			
20	Wf										T50		

III. Simulation of Distributed Engine Control with a Scheduled Network Model

Simulations were performed at sea level static conditions to test the proposed network schedule. The first set of simulations drastically reduced the messaging rates with the controller, as described in Table 2. These results are shown in Figure 3 through 5. Note that an engine pressure ratio (EPR) control is used for these simulations, and that the control setpoint is purposely not reached at low-power steady state. This was done to help observe how the engine transitions between control modes when scheduling logic is added. No attempt was made to modify the controller inputs; therefore, every change in the signal level results in a larger step-like change in quantization due to the lower sample rates. Observing the results, the gross system response of the scheduled and nominal trials track remarkably well. On closer observation, however, we see the controller tends to provide an underdamped command to the fuel flow actuator because of the step-like changes at its inputs during high slew times of burst (acceleration) and chop (deceleration). There are small variations in both spool speeds and the thrust response, which actually increased during acceleration. Most of these issues appear to be related to the controller riding limits due to delays in receiving sensor updates. There is no significant deviation in sensed temperatures, so these figures were not included.

Upon examining the time constants, it is observed that the change in shaft speed results in an unusually long time constant. It is anticipated that this would cause some problem with the controller because it ends up appearing as an excessively large step change. The existing controller does not provide smoothing, and we did not attempt to add it. Instead, we chose to increase the network communication rate to help minimize the step.

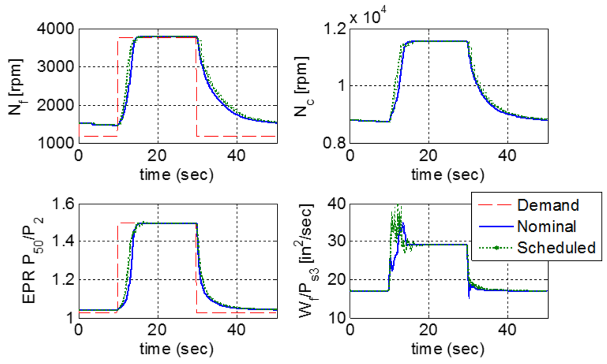


Figure 3. Simulation results after reducing messaging rates to the controller by 93% relative to the nominal full data rate. Shown are the low and high spool speeds, engine pressure ratio, and indication of fuel to air ratio under burst and chop conditions at sea level static conditions.

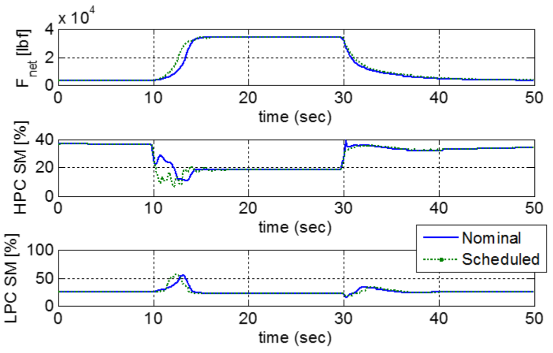


Figure 4. Simulation results with messaging rates reduced by 93%. The thrust acceleration response is actually slightly improved with some loss of stall margin in the high pressure compressor during acceleration. Sea level static conditions.

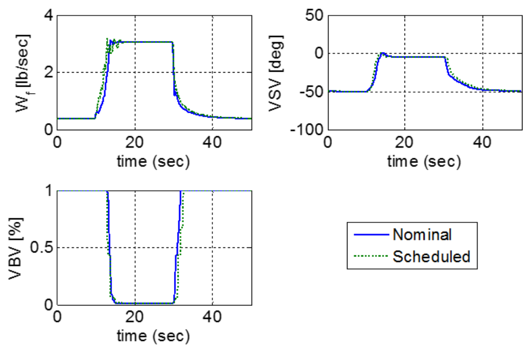


Figure 5. Actuator response shown under messaging rates reduced by 93%. Fuel mass flow rate, variable stator vane position, and variable bleed valve position for burst and chop profile at sea-level-static

A second simulation was run using a faster update rate on the shaft speeds only. They were increased by a factor of 5. This is exactly the same schedule shown in Table 4, except the periodic interval is now 10 for Nf and Nc. Even at the faster update rate, these sensors were still being sampled no faster than the highest sampling rate in the system. The response showed better agreement with the baseline.

For the second trial, the same plots are reproduced in Figure 6 through 8. The underdamping of the fuel flow command is significantly reduced. The tracking variation is only slightly perceptible, and is most prominent during deceleration. There are only minor differences in stall margin, relative to the nominal case.

Figure 9 is intended to provide an indication of the effect on the controller, due to the limited messaging, as it attempts to respond to the burst and chop. The throttle command is shown in the bottom plot. The controller attempts to execute the throttle command to provide the requested thrust. It is normal for specific limit conditions, like acceleration/deceleration rates, temperature, or speed to be activated during these transient events in order to provide safe operation and preserve engine life.

The nominal engagement of these limiters in this engine system simulation is shown in the upper plot of Figure 9. The vertical lines indicate when the limiters are switching on or off. This happens very quickly, at 0.015 second intervals in this case, so the thicker the vertical lines appear the more switching is occurring between various limit conditions. The design of the controller determines which of these control modes is of the highest priority. When the controller is switching modes often and rapidly, it is called chattering. In the nominal case, limit conditions are engaged during the entire acceleration transient, from t = 10 seconds to 15 seconds, and at the beginning of the deceleration transient at t = 30 seconds. However, a very limited amount of chattering occurs.

In contrast, the middle plot shows what happens to the same controller when the network restricts the information to a lower periodic rate. During the acceleration, extensive chattering between various limit conditions occurs and the duration of the controller operating under limit conditions is extended by more than a second. Deceleration exhibits an even more pronounced effect as the duration of the limit extends for about 4 seconds and is under almost continuous chatter.

Using the time constant as the periodic rate of network data transfer to the controller guarantees that the frequency content of the original signal is preserved. However, this presents a similar effect as what occurs when digitizing a continuous time signal. Additional high frequency content is included in the information because the value is held constant until the following sample, resulting in a step change between each sample. The control law interprets the step change as a small

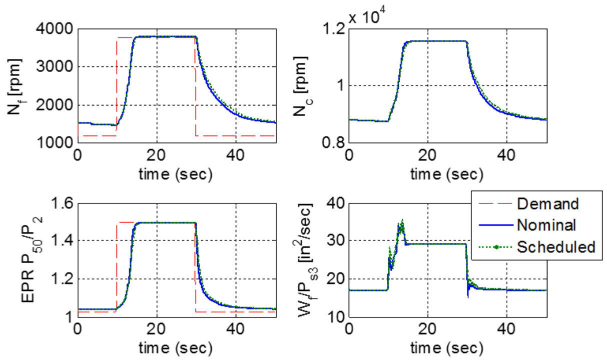


Figure 6 Simulation results using slightly more frequent communication of spool speeds compared to Figure 3 under the same conditions. The plot of Wf/Ps3 is much closer to nominal.

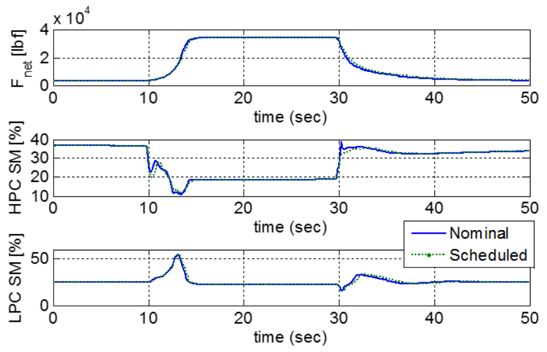


Figure 7 Net thrust and stall margin performance are greatly improved relative to Figure 4. This example communicates the spool speeds at a periodic rate 10 times slower than the controller interval.

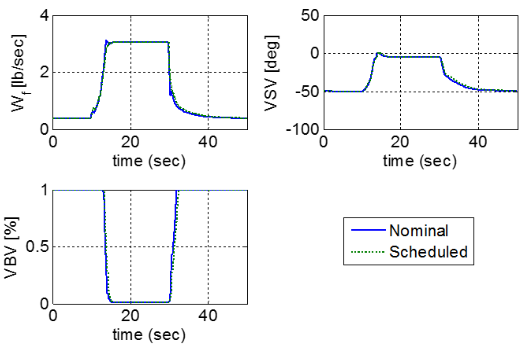


Figure 8 The fuel flow actuator response is significantly improved relative to Figure 5.

disturbance that can trigger a limit in the controller. It is assumed that these effects could be easily compensated for in the controller.

Clearly for the C-MAPSS40k system under consideration, the data rates to and from the controller can be greatly reduced without compromising the stability of the engine system. This is a promising initial result for a limited test case. However, additional evaluation is necessary to demonstrate that the results hold for more complex engine systems under more rigourous test conditions. Distributed engine control requires the use of serial communication networks. These networks will have at least some exposure to the engine's high temperature environment, as a result they are likely to be a more bandwidth limited resource relative to other ground-based control applications, due to electronics derating

While the results shown here are promising, it is not incumbent on the network and data transfer rates to maintain the engine system performance. The controller inputs should, at the very least, apply some smoothing to the sensor input data using knowledge of the throttle command and sensor response characteristics to minimized the step change that was shown to affect our results.

The very nature of sequential data transfer over serial networks results in a phase shift between when each sensor input is received at the controller. The controller has no ability to distinguish this phase shift when multiple data items are received within a single control interval, unless that data has a time stamp, implying the system is being synchronized. In our example, we assumed that all the data transferred to the controller is fresh, that is, it has a minimal lag between when it was requested and when it was received at the controller. Still, our schedule implemented a condition where it was rare for more than one fresh input to be received by the controller during a single interval. It may be beneficial to order the data in a certain way, or to acquire certain data in the same interval, but no attempt was made to determine this in our example.

Considering that the controller has the most complete information about the system, it is not unreasonable for the control law to take additional actions. Knowing that a throttle movement is occurring, or some other system transient is underway, the controller could temporarily reprioritize the sample rate of certain key variables. This would help prevent the chattering of control modes and could improve system performance.

Another alternative is to use model based techniques. The input from any sensor should be well known as a first or second order system. Model-based knowledge can be used to anticipate the controller input response characteristics, using the model value as the actual control input, and using the sampled data as model verification. All of this is the basis for future work.

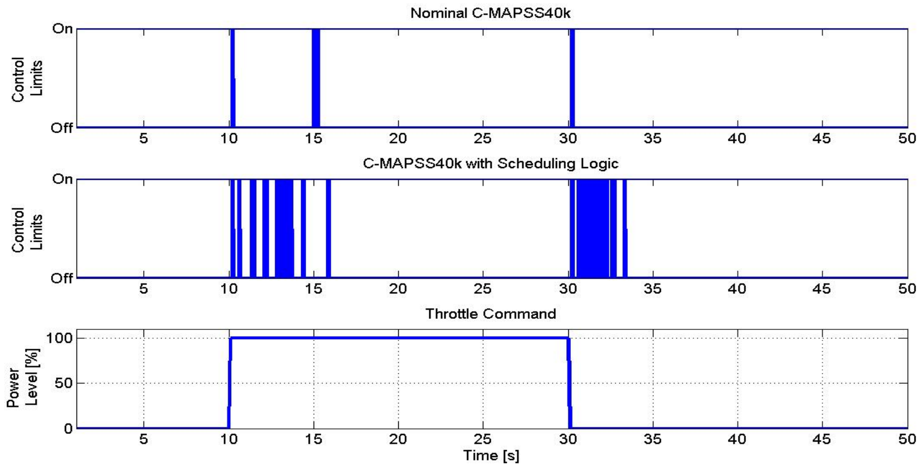


Figure 9. **A comparison of the effect of network communication on the operating mode of the controller. The bottom plot represents the throttle command for full power burst and chop. The top plot describes the nominal controller as limits are engaged during the transients. A small amount of chattering is observed. The middle plot describes the increase in chattering and extended duration of the same controller during the second trial. Here the messaging between controller and the sensors and actuators was reduced by 91.8% from the nominal case.**

IV. Conclusions

A turbine engine system simulation employing a distributed engine control architecture was analyzed for performance impact. The distributed control architecture is significantly different from a traditional centralized architecture because sensors and actuators are assumed to exist as independent smart nodes with embedded electronics. The controller employs serial digital networks to communicate information between itself and the smart node hardware to enact closed-loop control. The serial networks require that data flowing between these control elements occur at different times, that is, sequentially and not simultaneously, as may occur with a centralized control architecture. Furthermore, the network bandwidth may be limited in such a way that it is impossible to transfer all the system data to and from the controller because of bit rate, amount of data, and a limited control interval time. At best, attempting to transfer all system data during each control loop interval will result in a significant potential for race conditions between data transfer and control law evaluation. This study examined the potential impact of the sequential flow of control information on the turbine engine system performance. A network schedule model for nominal (not fault conditions) was developed that restricts the communication of information, from sensor to controller, and controller to actuator, based on a reasonable physical criteria. This criterion is the characteristic response of sensor and actuator devices in the control system assuming a simple first order response and the time constants of the individual devices. The assumption being that a centralized control architecture greatly oversamples the control data and common modeling tools inherently replicate this. Judiciously reducing the oversampling rate does not significantly affect the frequency content in the information communicated between the engine and the control law.

Using the communication scheduling model so developed, we successfully controlled the rate of information flow within the control system without affecting the fundamental update rate of the individual sensor or actuator models, which also interface to the engine plant. This is important for future distributed control technologies that may employ wide-bandwidth control locally on an engine using the processing capabilities of a smart node. The communication scheduling model rate for each device is individually selectable and easily updated. The results demonstrated that even at the lowest data rate, based on the time constants of each device, a reasonable and stable engine system performance was achieved. This rate reduction was an approximate 90% turndown in system messaging to and from the controller. Small deviations in performance were noted and several suggested actions to compensate for these deviations at the controller were suggested. Overall, we find that the change in how control information is communicated in a distributed engine control architecture is not likely to have detrimental effects on the performance, stability, or safety of future aero-propulsion systems. However, it is necessary to fully understand these effects by incorporating them in the modeling and simulation of such systems. Additional work will be required to rigorously and mathematically defend the empirical simulations provided here.

Acknowledgments

This work is being performed at NASA Glenn Research Center under the Transformative Aeronautics Concepts Program, Transformational Tools and Technologies Project.

References

[1] Jaw, L., Mattingly, J.D., 2009, Aircraft Engine Controls: Design, System Analysis, and Health Monitoring, American Institute of Aeronautics and Astronautics, Inc., pp. 1-35.

[2] Lewis, T. J. "Distributed architectures for advanced engine control systems." Advanced Aero-Engine Concepts and Controls 8 (1996).

[3] Schley, W. R., "Distributed Flight Control and Propulsion Control Implementation Issues and Lessons Learned," J. Engineering for Gas Turbines and Power, Vol. 121, No. 1, January 1999, pp. 96-101.

[4] Culley, D., "Transition in Gas Turbine Control System Architecture: Modular, Distributed, and Embedded," ASME Turbo Expo2010: Power for Land, Sea, and Air, Vol. 3, Glasgow, Scotland, United Kingdom, June 2010, pp. 287–297.

[5] Berner, A. W., "Challenges for Implementing a Distributed-Control System for Turbine Engines," ASME Turbo Expo2010: Power for Land, Sea, and Air, Vol. 3, Glasgow, Scotland, United Kingdom, June 2010, pp. 401-407.

[6] Culley, D.E., Paluszewski, P.J., Storey, W. and Smith, B.J., 2009. The case for distributed engine control in turbo-shaft engine systems, NASA TM-2009-215654, Glenn Research Center, Cleveland, Ohio September.

[7] Behbahani, A., Culley, D., Carpenter, S., Mailander, B., Hegwood, B., Smith, B., Darouse, C., Mahoney, T., Quinn, R. and Battestin, G., 2007. Status, Vision, and Challenges of an Intelligent Distributed Engine Control Architecture (Postprint) (No. AFRL-RZ-WP-TP-2008-2042). Air Force Research Lab Wright-Patterson AFB OH Turbine Engine Div.

[8] Neudeck, P.G., Okojie, R.S. and Chen, L.Y., 2002, "High-temperature electronics-a role for wide bandgap semiconductors?", Proceedings of the IEEE, 90(6), pp.1065-1076.

[9] Johnson, R.W., Evans, J.L., Jacobsen, P., Thompson, J.R. and Christopher, M., 2004, "The changing automotive environment: high-temperature electronics," Electronics Packaging Manufacturing, IEEE Transactions on, 27(3), pp.164-176.

[10] Amalu, E.H., Ekere, N.N. and Bhatti, R.S., "High temperature electronics: R&D challenges and trends in materials, packaging and interconnection technology," In Adaptive Science & Technology, 2009. ICAST 2009. 2nd International Conference on (pp. 146-153). IEEE.

[11] Cressler, J.D. and Mantooth, H.A. eds., 2012. Extreme environment electronics. CRC Press.

[12] J. K. Yook, D. M. Tilbury and N. R. Soparkar, "Trading computation for bandwidth: reducing communication in distributed control systems using state estimators," in IEEE Transactions on Control Systems Technology, vol. 10, no. 4, pp. 503-518, Jul 2002. doi: 10.1109/TCST.2002.1014671

[13]Yonggang Xu and J. P. Hespanha, "Estimation under uncontrolled and controlled communications in Networked Control Systems," Proceedings of the 44th IEEE Conference on Decision and Control, 2005, pp. 842-847. doi: 10.1109/CDC.2005.1582262

[14] Hespanha, J.P., Naghshtabrizi, P. and Xu, Y., 2007. A survey of recent results in networked control systems. PROCEEDINGS-IEEE, 95(1), p.138.

[15] Yedavalli, R.K., Zein-Sabatto, M.S., and Behbahani, A.R., "Framework for Distributed Engine Control System for Sampled-data Systems with Uncertain Time-varying Sampling Intervals and Delays with State Estimations," 51st AIAA Joint Propulsion Conference, AIAA-2015-3990

[16] May, R.D., Csank, J., Lavelle, T.M., Litt, J.S., and Guo, T.H., "A High-Fidelity Simulation of a Generic Commercial Aircraft Engine and Controller," AIAA–2010–6630, AIAA 46th Joint Propulsion Conference and Exhibit, Nashville, TN, July 25–28, 2010.

[17] Csank, J., May, R.D., Litt, J.S. and Guo, T.H., "Control design for a generic commercial aircraft engine," 46th AIAA Joint Propulsion Conference, July 2010, Nashville, TN, AIAA-2010-6629

[18] Zinnecker, A.M., et al, "A Modular Framework for Modeling Hardware Elements in Distributed Engine Control Systems," 50th AIAA Joiunt Propulsion Conference, July 2014,

[19] Thomas, G.L., et al, "The Application of Hardware-in-the-Loop Testing for Distributed Engine Control," 52nd AIAA Joint Propulsion Conference, July 2016, (submitted for publication)

[20] Aretskin-Hariton, E.L., "Design and benchmarking of a network-in-the-loop simulation for use in a hardware-in-the-loop system," AIAA Science and Technology Forum (SciTech 2017), (submitted for publication)